DOCUMENT RESUME

ED 062 158 SE 013 604

TITLE	Recommendations for an Undergraduate Program in Computational Mathematics.
INSTITUTION	Committee on the Undergraduate Program in Mathematics, Berkeley, Calif.
SPONS AGENCY	National Science Foundation, Washington, D.C.
PUB DATE	May 71
NOTE	49p.
AVAILABLE FROM	CUPM, P. O. Box 1024, Berkeley, California 94701 (Free)
EDRS PRICE	MF-$0.65 HC-$3.29
DESCRIPTORS	*College Mathematics; Computers; *Computer Science Education; *Curriculum; Mathematical Applications; Mathematics Education; Undergraduate Study
IDENTIFIERS	*Computational Mathematics; CUPM

ABSTRACT

This report describes an undergraduate program designed to produce mathematicians who will know how to use and to apply computers. There is a core of 12 one-semester courses: five in mathematics, four in computational mathematics and three in computer science, leaving the senior year for electives. The content and spirit of these courses are described, with detailed outlines and sample exercises for the computational mathematics courses (entitled, in order: Computational models and problem solving, Introduction to numerical computation, Combinatorial computing, and Differential equations and numerical methods). Recommendations are made on the implementation of the proposed program. (MM)

Recommendations for an UNDERGRADUATE PROGRAM in COMPUTATIONAL MATHEMATICS

A Report
of the
Panel on Computing

COMMITTEE ON THE UNDERGRADUATE
PROGRAM IN MATHEMATICS

May, 1971

RECOMMENDATIONS

FOR AN

UNDERGRADUATE PROGRAM

IN

COMPUTATIONAL MATHEMATICS

A Report of the Panel on Computing

COMMITTEE ON THE UNDERGRADUATE PROGRAM

IN MATHEMATICS

May, 1971

The Committee on the Undergraduate Program in Mathematics is a committee of the Mathematical Association of America charged with making recommendations for the improvement of college and university mathematics curricula at all levels and in all educational areas. Financial support for CUPM has been provided by the National Science Foundation.

Additional copies of this report may be obtained without charge from CUPM, Post Office Box 1024, Berkeley, California 94701.

COMMITTEE ON THE UNDERGRADUATE PROGRAM
IN MATHEMATICS

Alex Rosenberg, Chairman
Cornell University

Grace E. Bates
Mount Holyoke College

Ralph P. Boas
Northwestern University

D. W. Bushaw
Washington State University

Llayron L. Clarkson
Texas Southern University

Earl A. Coddington
University of California,
Los Angeles

Ronald L. Graham
Bell Telephone Laboratories

I. N. Herstein
University of Chicago

John W. Jewett
Oklahoma State University

John L. Kelley
University of California,
Berkeley

Donald L. Kreider
Dartmouth College

William J. LeVeque
Claremont Graduate School

Werner C. Rheinboldt
University of Maryland

Arnold E. Ross
Ohio State University

Dorothy Stone
University of Rochester

Maynard Thompson
Indiana University

James H. Wells
University of Kentucky

June Wood
South Texas Junior College

E. G. Begle
School Mathematics Study Group
(Ex Officio)

Victor L. Klee, President
Mathematical Association of
America
(Ex Officio)

Paul T. Mielke
Executive Director

Andrew Sterrett
Associate Director

John T. White
Associate Director

Katherine B. Magann
Administrative Assistant

PANEL ON COMPUTING

Werner C. Rheinboldt
University of Maryland
Chairman

Richard H. Balomenos
University of New Hampshire

Samuel D. Conte
Purdue University

Lincoln K. Durst*
American Mathematical Society

Herbert J. Greenberg*
University of Denver

Franz E. Hohn
University of Illinois

Thomas E. Hull
University of Toronto

Milton E. Rose
University of Denver

*Terms expired January 31, 1971.

The Panel is grateful for the assistance it has received from consultants who attended its meetings and advisers who reviewed parts of its work.

Consultants

Dorothy L. Bernstein
Goucher College

Robert Morris
Bell Telephone Laboratories

Jurg Nievergelt
University of Illinois

Robert J. Thomas
DePauw University

Maynard Thompson
Indiana University

Advisers

Victor R. Basili
University of Maryland

Derek G. Corneil
University of Toronto

Robert E. Lynch
Purdue University

Victor B. Schneider
Purdue University

David B. Wortman
University of Toronto

TABLE OF CONTENTS

		Page
	Preface	
1.	Philosophy and Aims of the Program	1
2.	Recommendations and Brief Course Descriptions	4
	2.1 Basic Component	4
	2.2 Elective Component	10
3.	Implementation of the Program	18
	3.1 Staff	18
	3.2 Facilities	18
4.	Detailed Course Outlines	22

PREFACE

During the last two decades the development of computers has helped to stimulate the dramatic increase and diversification in the applications of mathematics to other disciplines. In the belief that the time is appropriate for a systematic approach to the impact of computers on undergraduate mathematics programs, the CUPM Panel on Computing presents this report.

> Our basic recommendation is that mathematics departments should experiment with innovative undergraduate mathematics programs which emphasize the constructive and algorithmic aspects of mathematics, and which acquaint students with computers and with the uses of mathematics in computer applications.

A specific undergraduate program in computational mathematics is proposed. This is not a program in computer science, nor is it a minor modification of the traditional undergraduate mathematics major. It is, rather, a program in the mathematical sciences that combines courses in mathematics, computer science, and computational mathematics. It can be used as a basis for further specialization in any of several areas, including computer science, or mathematics, or one of the areas of application of mathematics.

1. Philosophy and Aims of the Program

In 1964 CUPM published a report entitled "Recommendations on the Undergraduate Mathematics Program for Work in Computing." "Impressed with the all-pervading influence of modern high speed automatic computing and with the need for well-trained people in this field,"[1] the Committee attempted in that report to set forth guidelines concerning "the proper training of such people, particularly from the standpoint of mathematics." The report was directed to "the combined group of mathematicians and computer scientists" in the "university or college organization charged with the responsibility for research and training in computer science" as well as in mathematics. In a sense, it was the first comprehensive statement about the undergraduate education of computer scientists.

Since publication of the above report, computer science has developed as a separate area of study. More and more colleges and universities are establishing computer science departments, and the number of students enrolled in computer science programs is increasing rapidly. The need for separate curriculum studies in this new area was recognized by the Association for Computing Machinery, and in 1968 its Curriculum Committee on Computer Science published a report entitled "Curriculum 68, Recommendations for Academic Programs in Computer Science."[2] This widely acclaimed report is still regarded as giving a good description of curricula in computer science. Its recommended minimal mathematics preparation is about equivalent to that usually required of students in the physical sciences and engineering.

More recently, three trends have become noticeable. First, there appears to have developed a strong tendency on the part of computer science programs to minimize prerequisite requirements in traditional mathematics, particularly analysis, and also to underemphasize or even to disregard most areas of scientific computing. Second, many disciplines, including in particular the biological, social, and behavioral sciences, have become increasingly mathematical, giving rise to a need in these fields for expanded education in mathematics and in scientific computing. Finally, the computer has begun to have a direct effect upon mathematics courses themselves. New courses, particularly in computationally-oriented applied mathematics, are being introduced into many mathematics curricula, and traditional courses are being modified and taught with a computer orientation. As an example of the latter we cite only the teaching

---

1. Quotations are from the above-mentioned report, which is now out of print.

2. Communications of the ACM, Volume 11, Number 3, March 1968. Reprints are available for $1.00 per copy from ACM, 1133 Avenue of the Americas, New York, N. Y. 10036.

of calculus. Approximately 100 schools now offer a course in calculus using the text, Calculus, A Computer Oriented Presentation, published by the Center for Research in College Instruction in Science and Mathematics.[3] Other computer calculus projects were reported in the 1969 CUPM Newsletter, "Calculus with Computers," now out of print.

These three trends all indicate a need for the mathematics community to accept a responsibility for mathematical or scientific computing, and to broaden educational opportunities toward a more encompassing "mathematical science" in which students may explore the areas of overlap between pure and computational mathematics, as well as computer science. There is thus a need for innovative undergraduate programs which provide for a wide range of options, different opportunities for graduate study, and a variety of future careers.

A new view of mathematics as a mathematical science in the above sense raises many curricular questions, to which several CUPM panels have begun to address themselves. In particular, a need arose for reappraisal of the already-cited 1964 report. Such a reappraisal is desirable if for no other reason than that a large number of all undergraduate mathematics majors are likely to find themselves later in some computer-related field.

The present report is the result of such a reappraisal by the CUPM Panel on Computing. From the outset it was evident that the aims of this report should be different from those of the earlier work, since its intended audience is different. The present report does not address itself to the training of computer scientists. Instead, its concern is for the education of mathematicians who will know how to use and to apply computers. Programs in computational mathematics necessarily have different objectives than do programs in computer science.

In accordance with our previous remarks, the mathematics program presented here is intended to be a departure from the traditional undergraduate mathematics curriculum. It should not be regarded, however, as a replacement for that curriculum, but rather, together with it, as one of several equally valid options for students of the mathematical sciences. It should meet the needs of students who plan to enter careers in scientific computing or who wish to enroll in graduate programs in computationally-oriented applied mathematics. With some suitably selected options during the senior year, a continuation in many computer science graduate programs should be possible. With other options, a continuation in pure mathematics should also be possible. At the same time, several of the courses included in the program meet the mathematical needs of students in other disciplines and may also be appropriate for prospective secondary school mathematics teachers.

---

3. For further information, write to CRICISAM, Room 212 Diffenbaugh, Florida State University, Tallahassee, Florida 32306.

The program proposed here is presented in a spirit of open experimentation, not as a final product. In its design the Panel has been neither as conservative nor as radical as it might have been. For instance, a conservative approach might be to combine a list of suitable mathematics courses of a traditional nature with a complementary list of computer science courses. This is easily accomplished in an institution having both a mathematics and a computer science department, but it leads to a large number of required courses and provides for little or no interaction between the two parts of the program. At the other extreme stands a curriculum in which computing has been completely integrated with the mathematical material, either by the introduction of new courses or by the repackaging of old ones.

In designing its program the Panel has taken a path somewhere between the extremes indicated above. Several new computer-oriented mathematics courses are described here; at the same time, some standard computer science and mathematics courses are included and, in particular, no recommendations are made concerning the redesigning of standard mathematics courses, such as the calculus, to include computer use. Where they are available, such computationally-oriented basic mathematics courses could be ideal components of this program, but their definition still requires considerable study and experimentation. The Panel felt that such a study on its part would serve only to divert its attention from its main concern, namely, the description of a new curriculum in computational mathematics for the undergraduate mathematics major which can be implemented in many institutions without excessive cost or delay.

In this latter connection the Panel believes that its program can be offered even by smaller colleges having suitable access to educational computing equipment, with only modest additions to their mathematics staffs. More specifically, through the junior year, the new computationally-oriented mathematics courses recommended here number only four. These, together with the three basic and relatively standard computer science courses, could be handled by the equivalent of one mathematician interested in applied mathematics with an emphasis on computing and numerical analysis and one specialist in computer science. The remaining core courses can be taught by the other members of the mathematics department. Clearly, this small staff could offer only a few of the additional courses listed in this report as possible electives, but the Panel believes that even such a minimal program would be desirable for many students.

2. Recommendations and Brief Course Descriptions

For a major undergraduate program in Computational Mathematics we recommend a basic core curriculum of 12 one-semester courses: 5 in mathematics, 4 in computational mathematics, and 3 in computer science. We will refer to these courses in the sequel, respectively, by the symbols M1, M2, M3, M4, M5, CM1, CM2, CM3, CM4, C1, C2, and C3.

Each of the courses carries 3 credits; at the same time, it is desirable that some of the computer-oriented courses include a scheduled laboratory period for which additional credit may be awarded. As described below, this sequence can be handled in three years, leaving the senior year for electives, also set forth below.

2.1 Basic Component

Before describing the 12 courses in the Basic Component, it may be instructive to illustrate one way of imbedding them into the first three undergraduate years. In the chart opposite, arrows indicate the "prerequisite structure," i.e., the dependency of each course on those which precede it. Notice that two courses are recommended for each semester. Mathematical progress within the program is not different from that in standard programs. If the student wishes to switch to pure mathematics after sampling the 8 core courses of the first two years, it will be a simple matter for him to do so with no loss of mathematical pace. It should also be noted that of the CM and C courses, three are taught in the first semester and four in the second semester of each year. This part of the program could easily be handled by the equivalent of two teachers in a small college where multiple sections are unlikely.

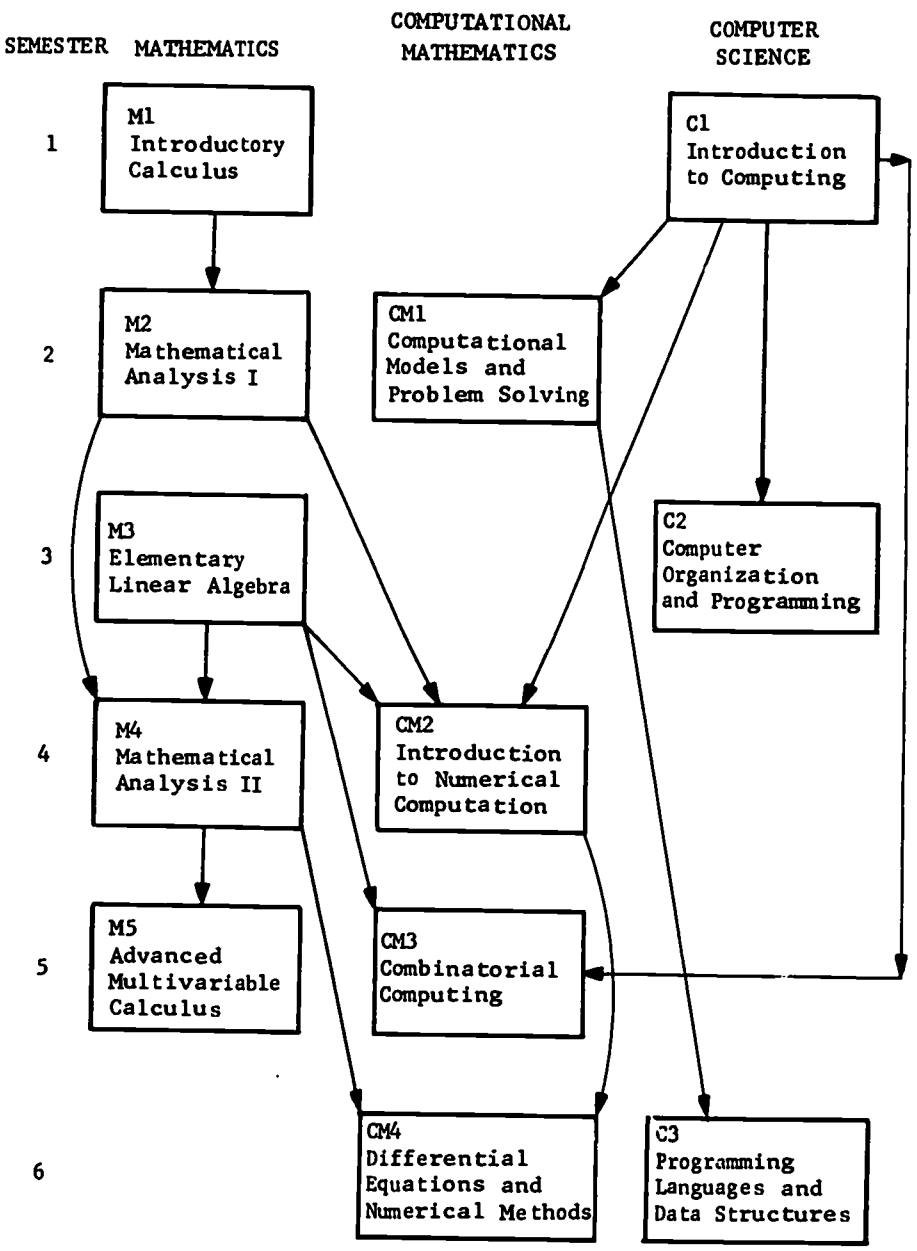
Let us now describe these 12 courses briefly, leaving detailed course outlines and references for Section 4.

a) Mathematics courses.

These five courses are described in the 1965 CUPM document, "A General Curriculum in Mathematics for Colleges," (GCMC), and in a forthcoming Commentary on GCMC. Incidentally, the Panel that is working on this Commentary has already noted that there is little need for M3 to require M1/M2 as explicit prerequisites. This fact has been observed in the chart on page 5.

M1	Introductory Calculus
M2, M4	Mathematical Analysis
M3	Elementary Linear Algebra
M5	Advanced Multivariable Calculus

CHART SHOWING ONE WAY OF IMBEDDING THE BASIC COMPONENT INTO THE FIRST THREE UNDERGRADUATE YEARS. ARROWS INDICATE THE PREREQUISITE STRUCTURE.



b) Computational Mathematics

These courses constitute the heart of our program. While their spirit is mathematical, computing plays an important role in each. The courses CM1 and CM3 are novel in character, while CM2 and CM4 are intended to replace the traditional first courses in Numerical Analysis and Ordinary Differential Equations. In the initial phase of implementing this program the traditional versions of these courses could be used temporarily in place of CM2 and CM4, thereby allowing the faculty to concentrate first on the development of the new courses CM1 and CM3.

CM1: Computational Models and Problem Solving

Prerequisite: C1

The purpose of this course is to introduce students early in their programs to a wide variety of different computer applications. This is to be accomplished mainly through the construction and interpretation of computational models for several interesting and worthwhile practical problems from various disciplines, including the biological and behavioral sciences as well as the physical sciences and mathematics.

The spirit in which the course is presented is of utmost importance. The applications discussed in the course should be reasonably realistic and comprehensive, and the students should become aware of the very serious difficulties and limitations that can arise. Questions should be raised about the validity of models, the effect of numerical errors, the significance of statistical results, the need for data verification, the difficulties in testing programs, documentation, etc. Wherever possible, the basic mathematical aspects of the different models should be discussed in general and related to the computational results. However, since the course is intended for freshmen or sophomores, no attempt can be made to enter into any deeper analysis of specific mathematical questions. With a proper balance between the computational and mathematical points of view, the course should not only provide the students with an appreciation of both the potential and limitations of computer applications, but also with an interest in learning more about the many relevant areas of mathematics.

The outline included in Section 4 places special emphasis on the use of computational models for the simulation of random and nonrandom processes, although a few numerical and nonnumerical computer applications are also included. The latter types of problem will be considered in more detail in the subsequent courses CM2 and CM3.

It should be noted that this course may also be of considerable value and interest to students outside the present program.

CM2: Introduction to Numerical Computation

Prerequisites: C1, M2, M3

This first course in numerical analysis may be taken in the sophomore year. Since it is based on as little as one year of analysis, the emphasis should be more on intuition, experimentation, and error assessment than on rigor. The methods considered should be amply motivated by realistic problems. It is better to treat a few algorithms thoroughly than to be exhaustive in the number of algorithms considered. Students should be expected to program and run a number of problems on a computer, and considerable time should be spent analyzing the results of such runs. In particular, the analysis of roundoff and discretization errors, as well as the efficiency of algorithms, should be stressed.

Topics should include the solution of linear systems, the solution of a single nonlinear equation, interpolation and approximation (including least squares approximation), differentiation and integration, and elements of the numerical solution of eigenvalue problems.

CM3: Combinatorial Computing

Prerequisites: C1 and M3

Combinatorial computing is concerned with the problem of how to carry out computations with discrete mathematical structures. It bears to combinatorial (discrete, finite) mathematics the same relationship that numerical analysis bears to analysis. Numerical analysis is much more widely known and much better developed than combinatorial computing. However, there are many reasons to believe that within the next decade combinatorial computing will rival numerical analysis in its importance to computer users. In fact, outside of the traditional areas of applications of mathematics to the physical sciences, discrete mathematical structures may occur more frequently than continuous ones, and even in large problems in the physical sciences data handling considerations lead quickly to questions in combinatorial computing.

This course is intended as an introduction to the emerging field of combinatorial computing. Its objectives are (1) to acquaint students with certain types of problems which occur frequently when problems are formulated in combinatorial terms, so that they are able to recognize them when they encounter them in disguise, and (2) to teach students certain important concepts and proven techniques which experience has shown to be useful in solving many combinatorial problems, particularly on a computer.

Typical topics to be covered in the course are the representation of integers, sets, and graphs; counting and enumeration techniques; sorting and searching methods; and selected problems and

algorithms in graph theory. Students should be expected to write programs for various algorithms and to experiment with their application to appropriate problems.

CM4: Differential Equations and Numerical Methods

Prerequisites: CM2 and M4

This course is intended to replace the more traditionally oriented course in differential equations in which the focus is often on non-constructive developments. It has the objective of introducing the student to key concepts underlying the qualitative understanding of differential equations as well as to methods for constructing their approximate solutions. It is intended for the junior year. The historical development of the subject is closely related to the physical and engineering sciences; nevertheless, it is recommended that examples from biology, economics, and other fields be chosen where possible, so as to draw upon a student's intuitive understanding of the processes illustrated. Some further suggestions for such material can be found in the CUPM report, "A Curriculum in Applied Mathematics" (1966).

As a result of this course the student should have confidence in his ability to develop an approximate solution of a differential equation, be able to discuss the basic qualitative behavior of the solution, and have an appreciation of the importance of analytic methods in furthering his understanding of the subject.

Typical topics should include a discussion of simple linear equations, the initial value problem for the first order equation $y' = f(x, y)$ and some methods for its numerical solution, a basic introduction to first-order systems and their applications including plane autonomous systems, and finally some topics relating to boundary value problems.

c) Computer Science.

The following three courses represent certain modifications of several of the basic courses in "Curriculum 68." All three courses should not consist simply of lectures, but should also incorporate a scheduled laboratory period.

C1: Introduction to Computing

Prerequisite: College admission

This first course in computing has by now become standard in many institutions. The earlier-cited 1964 CUPM report recommended a particular version of this course, and the corresponding course

B1 in "Curriculum 68" has been widely referenced. The course serves several purposes:

(1) To develop an understanding of the concept of an algorithm and of the algorithmic formulation of methods for the solution of problems on a computer.

(2) To train the student in the use of at least one algorithmic programming language and to introduce him to the basic structural aspects of such languages.

(3) To acquaint the student with the basic characteristics and properties of computers.

For the program proposed here the stress of the course should be on problem-solving by computer. Accordingly, the student should be assigned a number of different problems both of the numerical and non-numerical type, including at least one larger project.

C2: Computer Organization and Programming

Prerequisite: C1

The purpose of this course is to provide the student with a basic introduction to the structure and organization of digital computers and to the use of assembly language programming systems, without becoming involved in a too-detailed discussion of computer hardware or assembly language programming.

The course proposed here is in part similar to the course B2 in "Curriculum 68" with the addition of some topics from the course I3 in the same report. However, unlike those courses, it has primarily a survey character. Typical topics include computer structure, assembly languages, data representation, addressing techniques, elements of logic design, discussion of the principal units of a digital computer, systems software, and a survey of contemporary computers.

C3: Programming Languages and Data Structures

Prerequisite: CM1

This course is intended to introduce the student to some of the elements of programming languages as well as to certain important techniques of organizing and linking together information stored in a computer. Topics covered in the course include the basic structure of algorithmic languages, tree and list structures in a computer, string manipulation, data structure and storage allocation, and basic aspects of languages and grammars. The students should become acquainted with at least two different-level languages, such as a string manipulation language and an advanced algorithmic language.

The course covers a number of topics from the ACM courses I1 and I2 but is otherwise novel in character. Some instructors may find it desirable to use CM3 as a prerequisite; this would be similar in spirit to the approach of the ACM recommendations. But it is equally conceivable to introduce C3 as a prerequisite for CM3, allowing a much wider range of computational assignments in the latter course.

2.2 Elective Component

Given the Basic Component described above, and depending on the student's particular interests, there are several ways to round out a good major program. Broadly speaking, possible technical electives can be grouped under the following six--somewhat overlapping--categories, not necessarily in order of importance:

a) Mathematics
b) Probability and statistics
c) Computationally-oriented mathematics
d) Other applied mathematics
e) Computer science
f) Other disciplines

The specific courses listed here under each of these headings are not meant to exhaust all possibilities; clearly, there are various other choices and variations. If the Basic Component of the program has been completed during the first three years, the elective courses will--most probably--be concentrated during the senior and part of the junior year. But other arrangements of the Basic Component are also possible, thereby allowing for a distribution of elective courses throughout most of the undergraduate program.

a) Mathematics

Several of the courses offered as part of the standard mathematics curriculum can serve as electives for a computational mathematics program. This involves, in particular,

Introductory Real Variable Theory (GCMC 11, 12)
Complex Analysis (GCMC 13)
Algebraic Structures (GCMC 6)
Linear Algebra*
Introduction to Mathematical Logic

The Basic Component, augmented by a year course in real variables and a year course in algebra, would constitute minimally adequate preparation for graduate study in mathematics. These additions could easily be achieved in the senior year.

---

* Parallel with a course in algebraic structures, many departments also offer a course in linear algebra which covers the subject on a more advanced level than M3.

The standard introductory course in ordinary differential equations has not been mentioned here again since it was replaced by CM4. A beginning course in partial differential equations is included in subsection c) below.

b) Probability and Statistics

Statistical computations represent a large percentage of scientific computing work in many disciplines. Accordingly, the Panel believes that a good introduction to probability and statistics is highly important to students in a program of the kind discussed here. In fact, it may be very desirable to require such an introduction of all students in the program.

The courses on this subject included in the GCMC report do not appear to be entirely satisfactory. Instead, the Panel believes that a one-year combination of probability and statistics with M4 as a prerequisite is needed. The first semester could then provide an introduction to probability, with the second covering suitable topics from statistics. Courses like this are already offered in many schools, and recommendations about the material to be covered have been published by the CUPM Panel on Statistics.*

For the purposes of a computational mathematics program it may be highly desirable to integrate computational aspects directly into these courses. But in line with the approach taken in this report, the Panel did not wish to make any such specific recommendations at this time.

c) Computationally-oriented Mathematics

The courses grouped under this subheading are similar to CM1-CM4; that is, their spirit and content are mathematical, but computing plays an important role in each. Accordingly, it is most desirable that a program in computational mathematics include at least some additional courses of this nature.

From among the variety of possible topics the Panel decided to select five course areas which appear to be fairly representative.

Numerical linear algebra

This course covers the description and analysis of some of the principal computational methods in linear algebra. It uses CM2 and M3 as prerequisites and could replace the standard advanced linear algebra course for students in this program. Typical topics might

---

* "Preparation for Graduate Work in Statistics," 1971.

include a thorough discussion of elimination methods and of Wilkinson's backward error analysis, iterative methods for large linear systems and the corresponding basic convergence results, and methods for solving eigenvalue/eigenvector problems. The various topics should be motivated and illustrated by means of different applications.

Courses like this have become almost standard in many institutions. The course material can be found, for example, in parts of the following texts:

Forsythe, George E. and Moler, Cleve B. Computer Solution of Linear Algebraic Systems. Prentice-Hall, 1967.

Householder, Alston S. The Theory of Matrices in Numerical Analysis. Blaisdell, 1964.

Noble, Ben. Applied Linear Algebra. Prentice-Hall, 1969.

Varga, Richard S. Matrix Iterative Analysis. Prentice-Hall, 1962.

Wilkinson, James H. The Algebraic Eigenvalue Problem. Oxford University Press, 1965.

Applied modern algebra

The purpose of this course is to introduce the student to the discrete algebraic structures most commonly used in applications. It is intended to replace the standard modern algebra course (GCMC 6) for those students who are concerned with applications of algebra rather than with algebra as pure mathematics. Whereas the topics are in general not intended to be treated in depth, the treatment should be adequate enough in each case to enable the student to read independently in more complete expositions. Accordingly, the presentation should include formal definitions and proofs of fundamental theorems, but at the same time there should be considerable emphasis on practical applications.

While courses on applied and computational linear algebra have become reasonably common, the same cannot be said about courses on applied modern algebra. Moreover, at present, there exists essentially only one text on this topic, namely,

Birkhoff, Garrett and Bartee, Thomas C. Modern Applied Algebra. McGraw-Hill, 1970.

This book contains material for a full year course. A one-semester course on the senior level with a prerequisite of CM3 might begin with a review of set algebra and an introduction to semigroups and groups and some of their applications. Then the stress could be placed on partially ordered sets, lattices and Boolean algebras, and their applications in switching algebra and logic. Another approach would be

to play down Boolean algebra and to stress rings and fields, including, in particular, polynomial rings and finite fields, and their applications to coding theory.

Optimization

Optimization problems arise frequently in scientific computer applications. This includes problems from the entire area of mathematical programming as well as from optimal control theory, calculus of variations, and from parts of combinatorics. A one-semester introductory course in optimization problems, with CM2 and M4 as prerequisites, is therefore a highly desirable elective in a program of this kind.

Such a course--not stressing computational aspects--was described in the CUPM report, "Mathematical Engineering, A Five Year Program" (October 1966).* It begins with a discussion of specific examples of typical optimization problems from the various cited fields, and continues with an introduction to convexity and n-space geometry, Lagrange multipliers and duality, and the Simplex method. Then it turns to some combinatorial problems and to elements of the classical calculus of variations and of control theory. In a more computationally-oriented version of the course it appears to be desirable to delete the latter three topics and to present instead an extended coverage of the numerical aspects of linear programming, as well as a discussion of transportation problems. The course could then end with an introduction to numerical methods for convex programming problems. The student would be assigned computational projects involving some of the many available library subroutines; and in fact, an important by-product of the course in this form might be to familiarize the students with the extensive computational effort that has already been spent in connection with mathematical programming techniques.

There is an extensive list of available references relating to this course. Without attempting to be comprehensive, we mention only the following books:

Berge, Claude and Ghouila-Houri, A. Programming, Games and Transportation Networks. John Wiley, 1965.

Dantzig, George B. Linear Programming and Extensions. Princeton University Press, 1963.

Hadley, George F. Linear Programming. Addison-Wesley, 1962.

_____. Nonlinear and Dynamic Programming. Addison-Wesley, 1964.

---

* See also the CUPM report, "Recommendations on the Undergraduate Mathematics Program for Engineers and Physicists," January, 1967.

Künzi, Hans P., Tzschach, H. and Zehnder, C. Numerical Methods of Mathematical Optimization With ALGOL and FORTRAN Programs. Academic Press, 1968.

Polak, E. Computational Methods in Optimization. Academic Press, 1971.

Partial differential equations and numerical methods

The general aim of this course is to survey the standard types of partial differential equations, including, for each type, a discussion of the basic theory, examples of applications, classical techniques of solution with remarks about their numerical aspects, and finite difference methods. By necessity, most proofs of existence and uniqueness theorems and of the properties of the numerical methods are to be omitted.

A course of this kind--based on CM4 and M5--requires in general two semesters, and even then it will be very demanding of the students at the senior level. Typical topics include first order equations and the elements of the theory of characteristics for linear and quasi-linear equations, linear second order equations in two variables, classification, canonical forms, a discussion of the wave, diffusion, and Laplace equations, and a survey of some topics about other equations. For a description of a one-year course on partial differential equations--not stressing numerical methods--see also the CUPM report, "Mathematical Engineering, A Five Year Program."

There do not appear to be any entirely appropriate texts for this course. The following are some possible titles:

Ames, William F. Numerical Methods for Partial Differential Equations. Barnes and Noble, 1970.
Probably too difficult as a text for a first undergraduate course, but valuable as a reference for the course.

Berg, Paul W. and McGregor, James L. Elementary Partial Differential Equations. Holden-Day, 1966.
Elementary introductory text, but does not emphasize numerical methods.

Forsythe, George E. and Wasow, Woolfgang R. Finite Difference Methods for Partial Differential Equations. John Wiley, 1960.
Important reference for numerical methods.

Mitchell, A. R. Computational Methods in Partial Differential Equations. John Wiley, 1969.

Weinberger, Hans F. A First Course in Partial Differential Equations. Blaisdell, 1965.
Introductory text which places special consideration on physical applications.

Introduction to applied functional analysis

The purpose of this course is to present some of the basic material of elementary functional analysis as it is of use and importance in numerical and applied mathematics. With a prerequisite of CM2 and M5, typical topics included in the course would be an introduction to metric spaces, the contraction mapping theorem and various of its applications, normed linear spaces, linear and nonlinear operators, the differential calculus on normed spaces, applications to iterative processes, such as Newton's method, minimization techniques for nonlinear functionals on Banach spaces, and, if time permits, some discussion of the relationships between functional analysis and approximation theory.

By necessity, the material has to be presented from a geometrical and intuitive viewpoint rather than in a formal and abstract manner. Some of the results should be explored further by applying them to specific computational problems; here team projects may be very appropriate.

The following are some texts which cover parts of the material mentioned above:

Collatz, Lothar. Functional Analysis and Numerical Mathematics. Academic Press, 1966.
Survey of many of the interactions between the two fields.

Davis, Philip J. Interpolation and Approximation. Blaisdell, 1963.
For the connections to approximation theory.

Dieudonné, Jean. Foundations of Modern Analysis. Academic Press, 1969.
For the differential calculus on normed linear spaces.

Goffman, Casper and Pedrick, George. First Course in Functional Analysis. Prentice-Hall, 1965.
For much of the basic material; not numerically oriented.

Goldstein, Allen A. Constructive Real Analysis. Harper and Row, 1967.
For minimization methods.

Kantorovich, L. V. and Akilov, G. P. Functional Analysis in Normed Spaces. Pergamon, 1964.
Contains a detailed discussion of Newton's method.

Kolmogoroff, A. N. and Fomin, S. V. Elements of the Theory of Functions and Functional Analysis, Vol. I, Graylock Press, 1957.

Schechter, Martin. Principles of Functional Analysis. Academic Press, 1971.

In some of these courses it may be desirable to add a second semester in order to provide a more extended coverage of the material. This applies also to CM4 where a second semester is probably very desirable for many students.

In addition to the above courses, the Panel believes that students in a program such as this might benefit by being able to deepen their knowledge in graph theory and combinatorics beyond the material covered in CM3. A course in this area will be described by the CUPM Panel on Applied Mathematics in a forthcoming report.

d) Other Applied Mathematics Courses

As discussed in the beginning, a main aim of this program is to provide the student with a basic understanding of the application and use of computers in the solution of scientific problems. Accordingly, it will be most important that the student acquire a certain familiarity with at least some of the many applications of mathematics and with mathematical model building.

A number of suitable topics for several undergraduate applied mathematics courses are discussed in the CUPM report, "A Curriculum in Applied Mathematics." More specific outlines for some courses in physical mathematics are described in the CUPM report, "Mathematical Engineering, A Five Year Program." From this report we mention, in particular, the following courses:

ME3 Mechanics
ME8 Electromagnetics
ME9 Thermodynamics and Statistical Mechanics
OR2 Operations Research
OR3 Systems Simulation
OM3 Celestial Mechanics
OM4 Orbit Theory
CT2 Control
CT4 Linear Systems
CT7 Information Theory

It should be stressed that for the purposes of this program the specific topics covered in any of these courses are not as important as the applied mathematical spirit, that is, the emphasis on model building, on analysis of the model and on interpretation of the results.

It should also be noted that by listing these courses separately from those in the previous subsection we do not mean to imply that little or no computational work is to be involved here. In fact, in many of these courses computer applications might prove to be of considerable value and might strengthen the student's understanding of the interrelationship between scientific problems, mathematical models for them, and numerical methods for finding approximate solutions of these models.

e) Computer Science

In an institution with an ongoing computer science program, many of the courses offered as part of that program can serve well as possible technical electives in this curriculum. We mention, in particular, the following courses described in "Curriculum 68":

I4 Systems Programming
I5 Compiler Construction
I6 Switching Theory
I7 Sequential Machines
A1 Formal Languages and Syntactic Analysis
A2 Advanced Computer Organization
A4 System Simulation
A5 Information Organization and Retrieval
A7 Theory of Computability

Some introductory courses in mathematical logic covering topics from I7, A1, and A7 are also offered by many mathematics departments and may serve as possible electives.

Finally, it may be of interest to note that with the addition of three or four courses, such as I4 and I5 or A1, to the Basic Component, a student would meet more than the minimal requirements in "Curriculum 68" for an undergraduate major in computer science. Such additions could easily be achieved in the senior year.

f) Other Disciplines

This last and yet by no means least important subgroup of possible electives concerns courses in any of the disciplines outside of mathematics which are sources of mathematical computing problems. The Panel firmly believes that an understanding of the ideas, principles, and methods of at least one such area is a basic ingredient of the education of a computational mathematician and hence that any student in this program should take at least some suitable courses in another discipline. It should be stressed that this need not be the traditional introductory physics sequence, but that beginning courses in the engineering, biological, behavioral, or social sciences might be equally appropriate. The specific type and number of courses depends in each case on what is available, the field selected, and the student's depth of interest.

3. Implementation of the Program

3.1 Staff

It was stated earlier that the program of the Basic Component could be carried out by a mathematics department with the equivalent of one faculty member interested in numerical analysis and computing, and one in computer science. Such a department could offer some of the elective courses as well, depending on the interests of its members. A year course in Probability and Statistics is already taught in many colleges, and courses in Modern Applied Algebra are beginning to appear in addition to, or as replacements for, the usual courses in Abstract Algebra. Courses resembling those in Optimization or Applied Functional Analysis are also offered by many colleges.

Thus, while the entire program could not be offered except in an institution with several faculty members in applied and numerical mathematics as well as in computer science, much of it--and especially the Basic Component--may be possible in a small college with an expanded mathematics department as described above, provided a computer is available.

This leaves, of course, the question of staffing the computing facility itself, which in turn depends strongly on the nature of that facility. In most cases, such a facility requires the supervision of at least one professional manager or director, who in turn may be capable of teaching the necessary computer science courses in this program. Besides this person, many colleges have found that the problem of staffing the computing laboratory can be solved in part, or even completely, through the students themselves. One of the virtues of the computer as an instructional device is the personal involvement that it demands of and readily receives from the students. They learn quickly for the most part and teach one another very effectively. They serve well in many jobs associated with the operation of the computer facility. To bring them formally into the teaching process is sensible and rewarding.

3.2 Facilities

Apart from the arrays of desk calculators which have served statistical laboratories in the past, mathematics departments have not faced the wide variety of problems connected with the incorporation of laboratory work into their academic programs. The implementation of this program necessarily requires careful planning and maintaining of proper laboratory facilities. Because of sustained increases in costs of education, college administrations are understandably hesitant to incur major new expenditures. The following discussion is directed toward helping to clarify or distinguish among various factors which might characterize a computational facility suitable to this program.

The principal ways of incorporating computer use into an educational program can be characterized as follows:

1) Discussion of computational results obtained directly or indirectly by the instructor.

2) Student use of computers outside the classroom in a batch mode. Here, typically, programs are collected and submitted to be run together on a computer, without the possibility of further interaction from the originator. Very often the input is in the form of punched cards.

3) Student use of computers outside the classroom in a time-sharing mode. Here either simple teletypewriters or more elaborate character- and graphical-display devices are in open communication with the processor, and a user can input his program almost instantaneously and, by executing or modifying it at will, he is able to interact in an experimental manner with the computational process.

4) Use of time-shared classroom display facilities to integrate the presentation of the theoretical and computational aspects of the course material.

5) Use of special laboratories having dedicated computers (i.e., reserved solely for this use) for part or all of the meetings of the class in order to integrate computational work directly into the instructional process.

6) Use of special laboratories for computer-aided instruction.

At present, the most frequently used approaches are those under 1), 2), and 3), and, for this program, 1) by itself is not satisfactory. Accordingly, we shall focus our discussion primarily upon the use of the batch mode 2) or the time-sharing mode 3).

No matter which type of computational service is chosen, the most essential points appear to be that it must be reliable, responsive to fluctuating student demands during a semester, and capable of allowing the student to complete assignments in a reasonable time span. In line with this, a complete dependence on slack-hour use of a computer owned by local industry, the shared use of campus equipment dedicated primarily to accounting and administrative functions, or the "generous" gift of an outdated computer will generally prove unsatisfactory.

For most of the requirements of this program, computational services in the batch mode can be entirely satisfactory, effective, and at the same time economical. One of the critical factors is then the "turn-around time" between the submission of input and the return of the output to the originator. Since the completion of a problem by a student may require four to eight, or even more, machine runs, a turn-around time that allows at least two runs during a normal day appears to be rather desirable. (With the aid of multi-processing

systems, it is possible to achieve a turn-around time of a few minutes or less for short student runs.) Besides the turn-around time, another controlling parameter in batch service is the availability of ancillary equipment for producing and handling punched cards. Here queues easily develop which are not readily reduced without considerable cost. It may be hoped that this latter problem will be alleviated considerably by the development of less expensive mark-sensing or character-reading devices.

Time-sharing services have much to recommend them. However, their costs are generally higher than those of acceptable batch services. Moreover, they can also lead to considerable queueing problems if enough consoles are not available to the students. The critical parameter is the maximal number of terminals which can be sustained by the particular computer system without a significant degradation of the response time.

The repertoire of available computer languages is an important consideration for any computational service. For many of the requirements of this program one scientific language, such as FORTRAN, BASIC, ALGOL, PL/1, or APL is sufficient. In general, however, it is desirable that the student gain experience with more than one language, and in certain courses, such as C3 or CM3, additional languages such as SNOBOL are particularly important. In several courses, including, for instance, CM1, CM2, or CM4, plotting and display facilities could also play a useful role. Indeed, here a versatile time-shared classroom display system of the type mentioned under 4) above, might be ideal and could completely determine the character of the courses. However, more modest services can be completely successful.

Broadly speaking, the computational services required by this kind of program can be provided in one or a combination of the following ways:

1) Use of off-campus computing facilities;

2) Participation in an educational computer network;

3) Operation of a campus-wide educational computer facility;

4) Operation of separate computer laboratories by different departments.

Except under special circumstances, exclusive dependence on the first of these approaches is--in the long run--not very satisfactory. However, certain supplementary off-campus computer services, if reliable and economical, can provide highly advantageous solutions to enriching more modest services available on the campus.

At present, educational computer networks have been established in only a few geographical locations. The organization of these networks ranges from fairly loose mutual assistance groups, to highly-organized hardware networks. Either time-sharing or batch-

processing services can be provided--sometimes both. Clearly, the access to a large central computer with a massive program library, large memory, fast central processor, and large systems and programming staffs represents a considerable advantage. On the other hand, logistical and communication problems, lack of control, etc., may turn out to be very detrimental for a participant college. Nevertheless, where feasible, the possibility of joining such a network certainly deserves proper consideration.

Probably the most common approach toward meeting the educational computer needs of a college or university is the establishment of a centralized, campus-wide academic computing center. Such a center will serve its expected purpose only when operated by an adequate staff in an efficient, professional manner; this is a point too often overlooked.

In recent years many small and medium-sized computers have been marketed at relatively low prices. This has made it possible for many institutions to have separate computers of varying sizes for individual departmental use. The assured availability of a specialized service to the department is, of course, one of the greatest advantages of this approach. It also allows the development of special laboratories of the type mentioned under 5) above. On the other hand, the computational work possible on these machines is severely limited by their size, and for more sophisticated tasks additional computer services are often needed.

The actual costs of a computing facility depend upon many factors, including the desired quality of the service, the intended group of users, the specific type of equipment selected, local physical facilities, and the corresponding staff needs. The Panel therefore decided not to include here any cost estimates for the facilities needed in this program. Some data on such costs are given, for example, in recent reports of the Southern Regional Education Board and the American Council on Education.*

---

* See Guidelines for Planning Computer Centers in Universities and Colleges and Computers in Higher Education, both publications of SREB, 130 Sixth Street, N.W., Atlanta, Georgia 30313. See also Computers on Campus, American Council on Education, One Dupont Circle, Washington, D.C. 20036.

4. Detailed Course Outlines

In this section we present outlines for the seven courses CM1, CM2, CM3, CM4, and C1, C2, and C3. They are intended to suggest topics which might be included in these courses and should not be interpreted as check lists of required material. Where appropriate, suggested numbers of lectures to be spent on the various topics are included in the descriptions. These lectures total approximately 36 hours for a semester course; this leaves room for examinations, reviews, and lectures on supplementary technical material.

CM1: Computational Models and Problem Solving

Prerequisite: C1

Depth of treatment for the topics outlined below will vary with the interests of instructor and students, and lecture hours are therefore not assigned to any of the topics. It is recommended that a small number of fairly substantial projects be required in this course, rather than a larger number of smaller problems. Some of the material is suitable for group projects.

Detailed Outline

Statistical calculations

- Tabulation of data
- Calculation of means and variances
- Least squares fitting of straight lines
- Intuitive meaning of randomness
- Random number generators
- Tests of generators (e.g., chi-square)

Simulation of random processes

- Queues, inventories, random walks, etc.
- Discussion of statistical significance (confidence intervals)
- Games such as Blackjack and Bingo
- Monte Carlo calculations

Simulation of nonrandom processes

- Simple hypothetical computer
- Approximations to physical, economic and biological processes
- Discussion of errors in such approximations
- Deterministic games such as NIM

Other nonnumerical problems

- Enumeration
- Searching and sorting

Connectivity of graphs, shortest paths
Text editing
Elementary computer graphics
Handling arithmetic expressions

Sample Problems

1. Develop a program for the least squares fitting of straight lines to given data. The program should input pairs of values (x_i,y_i) and output the values of (a,b), where $y = ax + b$ is the best fit. Is the same line obtained when the values of x and y are interchanged? Show how your program can be used to fit curves given by $y = ab^x$, or $y = ax^b$, by taking logarithms of each side of these equations. Are the results the same as those obtained by a true least squares fit of these curves without taking logarithms?

2. Write a program to generate 1000 pseudo-random numbers and calculate the chi-square statistic that is associated with 10 equal sub-intervals of the interval in which the random numbers are supposed to be uniformly distributed. If the numbers are random, the value of this statistic should exceed 16.9 with a probability of only 5 percent. On the basis of this test, have you any reason for doubting the usefulness of your generator?

3. One relatively simple game of solitaire begins with a deal of nine cards, face up. If any two of these cards have the same face value, they are covered with two new cards, also face up. The last step is repeated until the deck has been exhausted, except for one card, in which case the dealer has won the game, or until there are no more pairs showing, in which case the dealer has lost. Write a program to simulate this game and use it to determine an approximation to the probability of winning. How reliable do you believe the approximation to be?

4. Describe a model of cars moving through a highway toll station, and write a program to simulate the process. Use it to find approximations to the average delay and show how this delay depends on traffic density. Discuss the main limitations of your model. Assuming that one has a good model, what further limitations are there in the results obtained from any such simulation?

5. Write a program to simulate a game of Blackjack and use it to compare different strategies. (This problem can be used as the basis for a group project.)

6. Describe a simple hypothetical computer and write a program to simulate its behavior. The description of the machine should be carefully documented so that any potential user will be able to determine exactly what the machine will do in every conceivable circumstance.

7. A man starts at the southwest corner of a field and runs north at 15 feet per second. His dog starts at the southeast corner, 200 feet from where the man starts, and runs directly towards his master at the rate of 40 feet per second. Calculate an approximation to the dog's path and to the time taken by the dog to catch his master. Compare this time with the time required if the shortest path had been taken.

8. Suppose that the adjacency matrix for a graph is given, along with two of its nodes. Write a program that will determine whether or not there is a path between the two nodes. Develop a second program for the same task, but based on a distinctly different algorithm, and compare the relative merits of the two different programs.

9. Develop a program for right-justifying text material. Input to the program should be a paragraph of text, and the corresponding output should be the same paragraph properly justified. (This problem can be expanded into a more substantial project on text editing by including additional features such as section headings and paging.)

10. A package of programs is to be developed for producing sequences of pictures. The pictures are to be output on a printer and must therefore be relatively simple, but the basic ideas are similar to those needed for computer produced movies. (This can be a good group project. Once agreement is reached on how to represent the data, members of the group can be assigned separate tasks, such as developing subprograms for input, output, moving, shrinking, and rotating pictures.)

Bibliography

Most introductory books on computer programming contain material on computer applications. Some of these texts are cited in the outline of the course C1 below. The following texts are primarily concerned with computer application problems suitable for this course:

Barrodale, Ian, Ehle, Byron L. and Roberts, F. D. K. Elementary Computer Applications. John Wiley, 1971 (to appear).

Gruenberger, Fred and Jaffray, George. Problems for Computer Solution. John Wiley, 1965.

Hull, Thomas E. and Day, David D. F. Computers and Problem Solving. Addison-Wesley (Canada), 1970 (in particular, Part 2).

CM2: Introduction to Numerical Computation

Prerequisites: C1, M2, M3

Each of the major topics in the course should be amply motivated by introducing applications from the physical and social sciences. A consideration of electrical networks or input-output systems in economics leads, for instance, to linear systems; vibration problems from mechanics, or Markov processes, provide examples for eigenvalue problems; root-locus problems arise in several areas of engineering; observational data collected in practical experiments lead to a consideration of interpolation and least squares techniques. A selection of problems can be found, for example, in the book by Carnahan, Luther, and Wilkes (see bibliography on page 26).

During the course the students should solve a number of problems on the computer. Some of these should involve programming of the simpler algorithms and others should make use of library subroutines.

Detailed Outline

Introduction (2 lectures)

- Number representation on a computer
- Computer arithmetic
- Discussion of the various types of errors

Linear systems of equations (9 lectures)

- Gaussian elimination and the LU factorization
- Partial and complete pivoting
- Example of ill-conditioning
- Discussion of ways for detecting ill-conditioning
- The Wilkinson backward error result and its implications (no proofs)
- Iterative improvement
- Iterative methods with simple convergence criteria (no proofs)

Solution of a single nonlinear equation (6 lectures)

- Successive approximation
- The Point of Attraction theorem and its implications
- Discussion of the rate of convergence
- Newton's method and the simplified Newton method
- Secant method and method of False Position
- Stopping criteria for iterations
- Extension of Newton's method to two equations in two unknowns
- Roots of polynomials
- Sturm sequences
- Example of ill-conditioning of the roots of a polynomial

Interpolation and approximation (6 lectures)

- Lagrange interpolating polynomial
- Error term for an interpolating polynomial
- Newton forward and backward difference polynomials
- Piecewise polynomial interpolation
- Least squares approximation, including numerical problems associated with the normal equations, and orthogonal polynomials and their use in least squares
- Chebyshev economization of power series

Numerical differentiation and integration (6 lectures)

- Error in differentiating the interpolating polynomial
- Differentiation by extrapolation to the limit
- Integration formulas based on interpolating polynomials and the associated error terms
- Romberg integration
- Gaussian quadrature formulas
- Adaptive methods

The eigenvalue problem (6 lectures)

- Direct root-finding methods such as Muller's or the Secant method
- The power method for the dominant eigenvalue
- Subdominant eigenvalues by the inverse iteration method
- The Householder-Givens method for symmetric matrices (without proofs)

Bibliography

Carnahan, Brice, Luther, H. A. and Wilkes, James O. Applied Numerical Methods. John Wiley, 1969.
Primarily as a source of problems.

Conte, Samuel D. Elementary Numerical Analysis. McGraw-Hill, 1965.
A second edition is in preparation.

Fox, Leslie and Mayers, D. F. Computing Methods for Scientists and Engineers. Oxford University Press, 1968.

Fröberg, Carl E. Introduction to Numerical Analysis, 2nd ed. Addison-Wesley, 1969.

Henrici, Peter K. Elements of Numerical Analysis. John Wiley, 1964.

McCracken, Daniel D. and Dorn, William S. Numerical Methods and FORTRAN Programming. John Wiley, 1964.

Stiefel, E. L. An Introduction to Numerical Mathematics. Academic Press, 1963.

Wendroff, Burton. First Principles of Numerical Analysis. Addison-Wesley, 1969.

CM3: Combinatorial Computing

Prerequisites: C1 and M3

The material listed here may be more than can be covered properly in one semester. Since many topics are rather independent of each other, an instructor can make his own selection of what to exclude. For this reason no breakdown into the number of lectures for each topic was included. Students are expected to implement some of the algorithms on the computer and also to experiment with relevant library subroutines. For this computational work, it may be desirable to assign team projects rather than to let every student proceed on his own.

Detailed Outline

The machine tools of combinatorics

- Integers and their representation, including radix-, modulo-, and factorial representation (and its use in indexing over permutations), monotonic vector representation (and its use in indexing over combinations and partitions)
- Sets and their representation, including bitstring and index representation
- Some aspects of list processing and storage organization, including representation of variable length sequences, one- and two-way lists, tree structures, free storage, and garbage collection

Enumeration and counting

- Enumeration techniques, such as backtrack and sieve methods
- Counting techniques, including recurrence relations and techniques for solving them, Polya's counting formula

Sorting

- Internal sorting, insertion-, selection-, and enumeration methods
- External sorting, long-sorted subsequences, merging, distribution sorting

Searching

- Searching in a linearly ordered set, including hash-coding or scatter storage techniques, Fibonacci search

Trees and their use in ordering sets, rooted trees and their properties, representation of trees, methods of traversing trees, internal and external path length, optimal and near optimal search trees
Heuristic search, game trees, minimax evaluation, pruning, static evaluation functions, backing up uncertain values

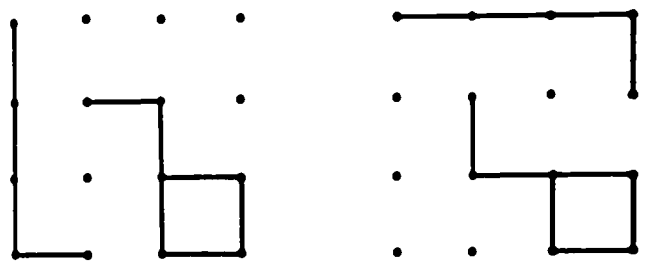
Graph algorithms

Some concepts from graph theory, such as graphs, directed graphs and their representation, paths, trees, circuits and cutsets
Connectedness and shortest path problems, including various related algorithms
Flow problems, max-flow and min-cut theorem, Ford-Fulkerson algorithm
Spanning trees, and algorithms for finding them
Graph isomorphisms
Planarity of graphs

Sample Problems

1. In how many different ways can one color the six faces of a cube which may be freely rotated with two colors? [Topics: counting, group of transformations, Polya's theorem]

2. An integrated circuit manufacturer builds chips with 16 elements arranged in a 4 x 4 array as shown below. To realize different circuits all patterns for interconnecting the elements are needed. Direct interconnections are made only between horizontally or vertically adjacent elements, e.g., as shown below:



(Closed loops do not usually occur, but this is ignored here for simplicity's sake.) To deposit interconnections on the chip a photo-mask of the interconnection pattern is needed. Notice that the same photo mask will do for the two interconnection patterns shown above. How many photo-masks are required in order to lay out all possible interconnection patterns on these chips?

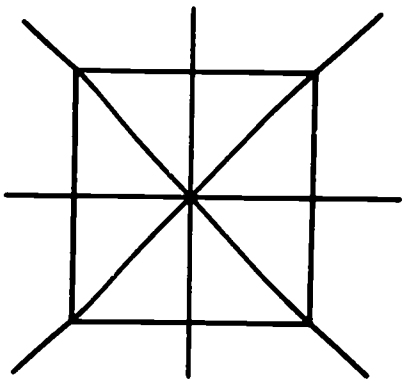
a) Carefully define the permutation group involved.

b) Solve the problem using Burnside's lemma alone.

c) Solve the problem using Polya's counting formula.

[Topics: counting, group of transformations, Polya's theorem]

3. List all the essentially different ways in which eight queens can be placed on a chessboard so that no two are on the same row, column, or diagonal. Two ways of placing queens are essentially different if they cannot be transformed into each other by a rotation of the board or by reflection on any of the axes shown in the figure:



4. Assume a large deck of N punched cards is dropped on the floor, but fortunately each card contains a unique sequence number from 1 to N which indicates its position in the deck. After the cards have been picked up, the deck is not in complete disorder; it contains long runs of cards in proper order. Discuss what sorting techniques can be considered to sort the deck as efficiently as possible. What standard sorting techniques would definitely be inefficient in this case? [Topics: linear order, expected number of comparisons, sorting algorithms]

5. a. Prove that every positive integer A has a unique factorial representation $a_1, a_2, \ldots, a_n$ which satisfies the conditions

(i) $A = a_1 \cdot 1! + a_2 \cdot 2! + \ldots + a_n \cdot n!$,

(ii) $0 \leq a_i \leq i$ for $i = 1, 2, \ldots, n$,

(iii) $a_n \neq 0$.

Let the factorial representation for zero be $a_1 = 0$, so that $0 = 0 \cdot 1!$.

b. Devise an algorithm for adding 1 to a number in factorial representation.

c. Devise algorithms for adding and subtracting two numbers in factorial representation.

d. For fixed $N \geq 1$, there are $(N+1)!$ numbers whose factorial representation $a_1, a_2, \ldots, a_n$ has $n \leq N$. From this fact and from the uniqueness of the factorial number representation proved in (a), derive the identity:

$$1 \cdot 1! + 2 \cdot 2! + 3 \cdot 3! + \ldots + N \cdot N! = (N+1)! - 1.$$

e. The factorial number representation is useful in enumerating permutations. This can be done in many ways. The technique discussed below is called the Derangement Method of M. Hall.

Let $P = (i_0, i_1, \ldots, i_N)$ be a permutation of the $N + 1$ integers $0, 1, \ldots, N$. For $j = 1, 2, \ldots, N$ define:

a_j = (the number of integers $< j$ which occur to the right of j in permutation P).

As an example, the permutation $P = (2, 0, 1)$ yields $a_1 = 0$, $a_2 = 2$.

By considering $a_1, \ldots, a_N$ to be the factorial representation of an integer A, we have set up a correspondence between the $(N+1)!$ permutations of the integers $0, 1, \ldots, N$ and the $(N+1)!$ numbers with factorial representation $a_1, \ldots, a_n$ $(n \leq N)$.

(e_1) Prove that this correspondence is 1:1.

(e_2) Devise an algorithm which constructs the permutation associated with an integer A from the factorial representation of A.

6. Devise an algorithm for finding shortest paths in a graph with weighted nodes. The length of a path is defined to be the sum of the weights of all nodes which lie on the path.

Consider the following three variations of the problem:

a. paths between two given nodes;

b. paths between one given node and all other nodes;

c. paths between all pairs of nodes.

[Topics: shortest paths, wave propagation algorithms]

Bibliography

There are several good books on combinatorial mathematics in general and on graph theory in particular, but there appears to be none which is written from the point of view proposed here, of emphasizing the computational aspects of algorithms for solving combinatorial problems.

The book that comes closest to this point of view is

Beckenbach, Edwin F. (ed.). Applied Combinatorial Mathematics. John Wiley, 1964.

Much useful material on computational and programming aspects of algorithms, combinatorial ones in particular, can be found in:

Knuth, Donald E. The Art of Computer Programming. Addison-Wesley.
Vol. 1, Fundamental Algorithms, 1968.
Vol. 2, Seminumerical Algorithms, 1969.
Vol. 3, Sorting and Searching, 1971-72.

The following references are not intended to be exhaustive by any means, but simply to point to a few papers which are typical of those which concentrate on computational aspects of combinatorics.

The machine tools of combinatorics

Hall, Marshall, Jr. and Knuth, Donald E. "Combinatorial analysis and computers." American Mathematical Monthly 72, Number 2, Part II(1965), pp. 21-28.

Lehmer, Derrick H. "The machine tools of combinatorics." Beckenbach, Edwin F. (ed.). Applied Combinatorial Mathematics. John Wiley, 1964.

_____. "Teaching combinatoric tricks to a computer." Proceedings of Symposia in Applied Mathematics 10. Combinatorial Analysis, pp. 179-194. American Mathematical Society, 1960.

Enumeration and counting

Golomb, Solomon W. and Baumert, Leonard D. "Backtrack programming." Journal of the Association for Computing Machinery 12(1965), pp. 516-524.

Lehmer, Derrick H. "The sieve problem for all-purpose computers." Mathematical Tables and Other Aids to Computation 7(1953), pp. 6-14.

Swift, J. D. "Isomorph rejection in exhaustive search techniques." Proceedings of Symposia in Applied Mathematics 10. Combinatorial Analysis, pp. 195-200. American Mathematical Society, 1960.

Walker, R. J. "An enumerative technique for a class of combinatorial problems." Proceedings of Symposia in Applied Mathematics 10. Combinatorial Analysis, pp. 91-94. American Mathematical Society, 1960.

Searching

Hibbard, Thomas N. "Some combinatorial properties of certain trees with applications to searching and sorting." Journal of the Association for Computing Machinery 9(1962), pp. 13-28.

Morris, Robert. "Scatter storage techniques." Communications of the Association for Computing Machinery 11(1968), pp 38-44.

Peterson, W. W. "Addressing for random access storage." IBM Journal of Research and Development 1(1957), pp. 130-146.

Graph algorithms

Corneil, D. G. and Gottlieb, C. C. "An efficient algorithm for graph isomorphism." Journal of the Association for Computing Machinery 17(1970), pp. 51-64.

Dijkstra, E. W. "A note on two problems in connexion with graphs." Numerische Mathematik 1(1959), pp. 269-271.

Edmonds, Jack. "Paths, trees and flowers." Canadian Journal of Mathematics 17(1965), pp. 449-467.

Gottlieb, C. C. and Corneil, D. G. "Algorithms for finding a fundamental set of cycles for an undirected linear graph." Communications of the Association for Computing Machinery 10(1967), pp. 780-783.

Lee, C. Y. "An algorithm for path connections and its applications." Institute of Radio Engineers Transactions on Electronic Computers EC-10(1961), pp. 346-365.

Moore, Edward F. "The shortest path through a maze." Proceedings of the International Symposium on the Theory of Switching, pp. 285-292. Harvard University Press, 1959.

Warshall, Stephen. "A theorem on Boolean matrices." Journal of the Association for Computing Machinery 9(1962), pp. 11-12.

CM4: Differential Equations and Numerical Methods

Prerequisites: CM2, M4

Throughout this course it is desirable to introduce problems which lead to the types of equations considered at the time. Excellent sources include circuit theory, mechanical systems, biological systems, particle dynamics, and economics. Numerical methods are to be introduced early in the course both to illustrate the qualitative behavior of solutions and to motivate uniqueness and existence arguments. In considering these methods the student should be made aware of the effects of discretization--and roundoff errors--and of stability. The students are expected to write some programs for various methods and to use existing library subroutines for others.

Detailed Outline

Origin and examples of differential equations (2 lectures)

- Sample (deterministic and non-deterministic) problems from the physical, social and biological sciences, including predator-prey model
- Difference equations, including examples of different equations leading to the same differential equation

Simple linear equations (4 lectures)

- $y' = f(x)$, $y' = ay + f$, $y'' = ay' + by + f$
- Representation of solutions by indefinite integrals and special functions
- Direction fields
- Qualitative behavior of solutions
- Uniqueness and continuous dependence on initial data
- Consequences of linearity
- Approximation by Taylor series
- Polygon method
- Trapezoidal approximation
- Equivalence of second-order equations to first-order systems
- Introduction to first- and second-order difference equations and their elementary properties

The first-order equation $y' = f(x,y)$ (9 lectures)

- Graphical treatment, polygon method
- Relation to integral equations, Picard iteration
- Quadrature methods
- Picard existence and uniqueness theorem with proof
- Statement of Peano existence theorem
- Non-uniqueness examples
- Discussion of continuous dependence on initial data
- Power series solution and numerical methods
- Runge-Kutta methods
- Predictor-corrector methods
- Discussion of consistency and convergence (without proofs)

First-order systems of equations (8 lectures)

Redevelopment for first-order systems--using vector notation--of the major results about single first-order equations
Review of matrix results, similarity transformations, series for exp(At) and semigroup properties
Vector space of solutions of $y' = Ay$, the adjoint solution
Representation of solutions of non-homogeneous problems
Stiff systems

Plane autonomous systems (7 lectures)

Numerical exploration of $y' = ax + by + f(x,y)$, $x' = cx + dy + g(x,y)$
Poincaré phase plane and critical solutions
Critical points and concepts of stability
Numerical comparison of linear and nonlinear equations
The Liensrd equations
Liapounov's ideas
Exploration of predator-prey model

Two-point boundary value problems (6 lectures)

Exploration of the linear second-order equation with mixed boundary conditions by shooting techniques
Discretization and methods for solving the resulting equations
Extensions to nonlinear equations

Bibliography

Birkhoff, Garrett and Rota, Gian-Carlo. Ordinary Differential Equations. Ginn, 1962.
Selected topics.

Daniel, James W. and Moore, Ramon E. Computation and Theory in Ordinary Differential Equations. W. H. Freeman, 1970.
For supplementary use.

Henrici, Peter. Discrete Variable Methods in Ordinary Differential Equations. John Wiley, 1962.

Keller, Herbert B. Numerical Methods for Two-Point Boundary Value Problems. Ginn College, 1968.
Advanced discussion of material on two-point boundary value problems.

Lapidus, Leon and Seinfeld, John H. Numerical Solution of Ordinary Differential Equations. Academic Press, 1971.

C1: Introduction to Computing

Prerequisite: College admission

As stated in Section 2, this course should be oriented toward problem solving with computers. Accordingly, it is important that, throughout the course, different types of problems are considered and appropriate algorithms for their computational solution are designed and discussed. In particular, it is essential that both numerical and nonnumerical applications are presented. The problems should be reasonably interesting and realistic, and some should be open-ended, requiring a certain effort to identify what is required and how the solution is to be obtained. At least one major project should be included leading to a completely verified and documented program.

The course can serve to introduce many traditional mathematical ideas from a different point of view (e.g., subroutines and functions, induction and recursion, etc.). Such identifications should be strengthened where possible.

The course should be organized so that students can write small computer programs almost immediately. This may be accomplished by representing algorithmic processes from the outset both by flowcharts and programming languages.

The following outline is for a one-semester course meeting three times each week for lectures. In addition, it is generally advisable to schedule a regular weekly laboratory period of at least two hours. No lecture hours were assigned since the need for proper sequencing of programming assignments often demands that certain topics are either interchanged or distributed throughout the course.

Detailed Outline

Problems, algorithms and programs

- Typical problems and mathematical models
- Concept of an algorithmic process
- Flow charts
- Basic structure and properties of algorithms
- Concept of a program
- How computers execute programs
- Elements of a higher-level programming language

Basic programming

- Number and character representation
- Constants and variables
- Principal syntactic statements of the language
- Functions, subroutines and complete programs
- Elements of the system being used

Libraries
Program testing and documentation

Errors and approximations

The approximative character of mathematical models
Truncation and round-off error
Verification of algorithms
Error conditions and messages
Techniques for algorithm testing
The idea of numerical stability

Data structures

Discussion of a variety of problems leading to different data structures such as vectors, arrays, strings, trees, linked structures
Basic manipulation of the different structures

Advanced topics

Further details of the programming language
Aspects of compilers
Basic structure of an operating system
Aspects and organization of computer systems

Survey of computers, languages, and systems

Historical developments, discussion of different language types, aspects of systems programs, new developments

Bibliography

Arden, Bruce W. An Introduction to Digital Computing. Addison-Wesley, 1963.
A good reference for the instructor.

Cole, R. W. Introduction to Computing. McGraw-Hill, 1969.

Forsythe, Alexandra I., Keenan, Thomas A., Organick, Elliott I. and Stenberg, Warren. Computer Science: A First Course. John Wiley, 1969.
This is a text for a high school course but may be appropriate for this course.

Galler, Bernard A. The Language of Computers. McGraw-Hill, 1962.
A good reference for the instructor.

Gruenberger, Fred. Computing: An Introduction. Harcourt, Brace and World, 1969.

Hull, Thomas E. Introduction to Computing. Prentice-Hall, 1966.

Hull, Thomas E. and Day, David D. F. Computers and Problem Solving. Addison-Wesley (Canada), 1970.
Part 1 of this text emphasizes material appropriate for this course.

Kemeny, John G. and Kurtz, Thomas E. Basic Programming. John Wiley, 1967.
An introduction to programming with applications.

Rice, J. K. and Rice, J. R. Introduction to Computer Science: Problems, Algorithms, Languages, Information and Computers. Holt, Rinehart and Winston, 1969.

Walker, Terry M. and Cotterman, William W. An Introduction to Computer Science and Algorithmic Processes. Allyn and Bacon, 1970.

C2: Computer Organization and Programming

Prerequisite: C1

This course includes computational projects in assembly language programming. However, in line with the survey character of the course, care should be taken not to involve the students in a too-detailed discussion of assembly languages or of computer hardware. A scheduled laboratory period is desirable.

Detailed Outline

Computer structure and machine language (2 lectures)

- Fundamentals of computer organization, including registers, arithmetic units, memory, I/O units and their interdependence
- Description of typical single-address machine instructions
- Programs as sequences of machine instructions and their execution

Introduction to symbolic coding and assembly systems (5 lectures)

- Mnemonic operation codes
- Labels, symbolic address
- Literals
- Pseudo operations
- General construction of assemblers
- Simple examples and exercises using a locally available assembler

Digital representation of data (3 lectures)

- Bits, fields, words
- Character representation
- Radix representation of numbers, radix conversion, representation of integers, floating point, and multiple precision numbers in binary and decimal form
- Variable length data

Addressing (2 lectures)

- Absolute addressing, indexing, indirect addressing, relative addressing
- Zero-, one-, two-, three-address instruction formats
- Address transformations
- Machine organization to implement addressing structures
- Character- versus word-oriented machines

Logic design (5 lectures)

- Elements of Boolean algebra
- AND, OR, NOT logic gates
- Implementation of Boolean functions
- Encoders and decoders
- Descriptive discussion of clocked circuits, flip-flops, registers, shift registers, accumulators, counters, timing chains

Arithmetic units (3 lectures)

- Serial versus parallel arithmetic
- Implications of choice of radix
- Design of a simple arithmetic unit
- Design of half-adder and adder
- Algorithms for multiplication and division

Instruction units (3 lectures)

- Instruction fetch and decoding
- Program sequencing
- Branching
- Subroutine calls
- Interrupts
- Control and timing logic
- Micro-programming as a means of implementing control units

Storage units (3 lectures)

- Structure of core memory
- Typical memory bus structure
- Memory overlap, protection, relocation and paging
- Word versus character organizations
- Types of bulk memories
- Descriptive discussion of stack memories, associative memories, read-only memories, and virtual memory schemes

Input-output systems (3 lectures)

Direct memory access I/O
I/O channels and controllers, multiplexers
Characteristics of various types of input/output devices
Relation of I/O system to control unit and main memory
Input/output programming
Buffering and blocking
Interrupts
Problems of error detection and correction in data transmission

Systems software (4-5 lectures)

Operating systems
Input/output packages
Assemblers, loaders
Interpreters, compilers
Utility programs and libraries

Survey of contemporary computers (3-6 lectures)

A survey of contemporary computers emphasizing a variety of machine organization. Typical topics: large versus small computers; single register, multiple register, and stack machines; unorthodox machines. Discussion of possible implementation of high-level programming language statements on typical computers.

Bibliography

Bell, C. G. and Newell, A. Computer Structures. McGraw-Hill, 1970.
Survey of computer organizations. Source of material for the sutvey of contemporary computers.

Chu, Yaohan. Digital Computer Design Fundamentals. McGraw-Hill, 1962.
A somewhat dated reference on logic design.

Gear, C. William. Computer Organization and Programming. McGraw-Hill, 1969.
Reference on assembly language programming.

Gschwind, H. W. Design of Digital Computers, An Introduction. Springer-Verlag, 1970.
Text on computer design and organization, slightly engineering-oriented.

Hellerman, H. W. Digital Computer System Principles. McGraw-Hill, 1967.
Uses Iverson notation, directed toward IBM equipment, especially S/360.

Knuth, Donald E. The Art of Computer Programming. Volume 2, Seminumerical Algorithms. Addison-Wesley, 1969.
Reference for a mathematical treatment of computer arithmetic (Chapter 4).

McCluskey, E. J. Introduction to the Theory of Switching Circuits. McGraw-Hill, 1965.
Reference for basic switching theory.

Nashelsky, Louis. Digital Computer Theory. John Wiley, 1966.
A paperback containing a survey of many of the topics covered in this course.

C3: Programming Languages and Data Structures

Prerequisite: C1

Detailed Outline

Structure of algorithmic languages (8 lectures)

- Review of basic program constituents of the language introduced in C1
- Introduction to the elements of ALGOL or PL/1
- Informal syntax and semantics of simple statements in that language
- Backus normal form
- Grouping of statements and block structure of programs
- Scopes, local and nonlocal quantities
- Functions and procedures
- Formal and actual parameters
- Binding time of program constituents
- Simple recursive procedures
- Concept of a stack
- Simulation of recursions as iterations using stacks

Arithmetic statements (4 lectures)

- Brief discussion of graphs and trees
- Tree diagrams of arithmetic expressions
- Informal discussion of precedence hierarchies
- Infix, prefix, postfix notation
- Translation between infix and postfix notation
- Evaluation of expressions in postfix notation

Trees and lists in a computer (8 lectures)

- Types of data nodes and linkages
- List names, list heads, sublists
- Multilinked lists
- Stacks as list structures with usage discipline
- Representation of trees as special cases of lists

Accessing, insertion, deletion and updating in trees
Traversal schemes for trees
Application to the generation of machine code from expression trees

String manipulation (7 lectures)

Introduction to a string manipulation language such as SNOBOL
Data declarations in such a language
Recursive algorithms in such languages
Applications to formal differentiation of expressions

Data structures and storage allocation (3 lectures)

Storage allocation for algorithmic language structures such as independent, nested blocks, strings, arrays, etc.
Procedures using run-time stacks
Storage allocation for string manipulation languages

Some aspects of languages and grammars (6 lectures)

Syntax, semantics and pragmatics of programming languages
The concept of a formal grammar
Production notation
Discussion of Chomsky's classification of grammars
Discussion of computability, undecidability
Syntax and semantics of arithmetic statements
Precedence and operator precedence grammars
Syntactic specification of procedures, blocks and statements
Formal semantics corresponding to syntactic specifications

Bibliography

Genuys, F. (ed.). Programming Languages. Academic Press, 1968.

Harrison, M. C. Data Structures and Programming. Courant Institute of Mathematical Sciences, New York University, 1970.

Knuth, Donald E. The Art of Computer Programming. Volume 1, Fundamental Algorithms. Addison-Wesley, 1968.
Important presentation of data structures.

Rosen, Saul (ed.). Programming Systems and Languages. McGraw-Hill, 1967.
Contains, among other things, a discussion of SNOBOL and a comparison of list processing languages.

Sammet, Jean E. Programming Languages: History and Fundamentals. Prentice-Hall, 1969.
A comprehensive survey of languages.

Wegner, Peter. Programming Languages, Information Structures and Machine Organization. McGraw-Hill, 1968.
An approach to programming languages as information structures.